Bitcoin and Cryptocurrency Technologies

Arvind Narayanan, Joseph Bonneau, Edward Felten,
Andrew Miller, Steven Goldfeder

with a preface by Jeremy Clark

Draft — Feb 9, 2016

Feedback welcome! Email bitcoinbook@lists.cs.princeton.edu

For the latest draft and supplementary materials including programming assignments, see our Coursera course.

The official version of this book will be published by Princeton University Press in 2016.
If you'd like to be notified when it's available, please sign up here.

Introduction to the book

There's a lot of excitement about Bitcoin and cryptocurrencies. Optimists claim that Bitcoin will fundamentally alter payments, economics, and even politics around the world. Pessimists claim Bitcoin is inherently broken and will suffer an inevitable and spectacular collapse.

Underlying these differing views is significant confusion about what Bitcoin is and how it works. We wrote this book to help cut through the hype and get to the core of what makes Bitcoin unique.

To really understand what is special about Bitcoin, we need to understand how it works at a technical level. Bitcoin truly is a new technology and we can only get so far by explaining it through simple analogies to past technologies.

We'll assume that you have a basic understanding of computer science — how computers work, data structures and algorithms, and some programming experience. If you're an undergraduate or graduate student of computer science, a software developer, an entrepreneur, or a technology hobbyist, this textbook is for you.

In this book we'll address the important questions about Bitcoin. How does Bitcoin work? What makes it different? How secure are your bitcoins? How anonymous are Bitcoin users? What applications can we build using Bitcoin as a platform? Can cryptocurrencies be regulated? If we were designing a new cryptocurrency today, what would we change? What might the future hold?

Each chapter has a series of homework questions to help you understand these questions at a deeper level. In addition, there is a series of programming assignments in which you'll implement various components of Bitcoin in simplified models. If you're an auditory learner, most of the material of this book is also available as a series of video lectures. You can find all these on our Coursera course. You should also supplement your learning with information you can find online including the Bitcoin wiki, forums, and research papers, and by interacting with your peers and the Bitcoin community.

After reading this book, you'll know everything you need to be able to separate fact from fiction when reading claims about Bitcoin and other cryptocurrencies. You'll have the conceptual foundations you need to engineer secure software that interacts with the Bitcoin network. And you'll be able to integrate ideas from Bitcoin into your own projects.

A note of thanks

We're immensely grateful to the students who helped develop programming assignments and to everyone who provided feedback on the drafts of this book. Princeton students Shivam Agarwal, Miles Carlsten, Paul Ellenbogen, Pranav Gokhale, Alex Iriza, Harry Kalodner, and Dillon Reisman, and Stanford students Allison Berke, Benedikt Bünz, and Alex Leishman deserve special praise. We're also thankful to Dan Boneh and Albert Szmigielski.

Preface — The Long Road to Bitcoin

The path to Bitcoin is littered with the corpses of failed attempts. I've compiled a list of about a hundred cryptographic payment systems, both e-cash and credit card based technologies, that are notable in some way. Some are academic proposals that have been well cited while others are actual systems that were deployed and tested. Of all the names on this list, there's probably only one that you recognize — PayPal. And PayPal survived only because it quickly pivoted away from its original idea of cryptographic payments on hand-held devices!

There's a lot to learn from this history. Where do the ideas in Bitcoin come from? Why do some technologies survive while many others die? What does it take for complex technical innovations to be successfully commercialized? If nothing else, this story will give you an appreciation of how remarkable it is that we finally have a real, working payment mechanism that's native to the Internet.

ACC	CyberCents	IKP	MPTP	Proton
Agora	CyberCoin	IMB-MP	Net900	Redi-Charge
AIMP	CyberGold	interCoin	NetBill	S/PAY
Allopass	DigiGold	Ipin	NetCard	Sandia Lab E-Cash
b-money	Digital Silk Road	Javien	NetCash	Secure Courier
BankNet	e-Comm	Karma	NetCheque	Semopo
Bitbit	E-Gold	LotteryTickets	NetFare	SET
Bitgold	Ecash	Lucre	No3rd	SET2Go
Bitpass	eCharge	MagicMoney	One Click Charge	SubScrip
C-SET	eCoin	Mandate	PayMe	TriVnet
CAFÉ	Edd	MicroMint	PayNet	TUB
CheckFree	eVend	Micromoney	PayPal	Twitpay
ClickandBuy	First Virtual	MilliCent	PaySafeCard	VeriFone
ClickShare	FSTC Electronic Check	Mini-Pay	PayTrust	VisaCash
CommerceNet	Geldkarte	Minitix	PayWord	Wallie
CommercePOINT	Globe Left	MobileMoney	Peppercoin	Way2Pay
CommerceSTAGE	Hashcash	Mojo	PhoneTicks	WorldPay
Cybank	HINDE	Mollie	Playspan	X-Pay
CyberCash	iBill	Mondex	Polling	

Table 1: Notable electronic payment systems and proposals

Traditional financial arrangements

Back in time before there were governments, before there was currency, one system that worked for acquiring goods was barter. Let's say Alice wants a tool and Bob wants medicine. If each of them happen to have what the other person needs, then they can swap and both satisfy their needs.

On the other hand, let's say Alice has food that she's willing to trade for a tool, while Bob, who has a tool, doesn't have any need for food. He wants medicine instead. Alice and Bob can't trade with each other, but if there's a third person, Carol, who has medicine that she's willing to trade for food, then it becomes possible to arrange a three-way swap where everyone gets what they need.

The drawback, of course, is coordination — arranging a group of people, whose needs and wants align, in the same place at the same time. Two systems emerged to solve coordination: credit and cash. Historians, anthropologists, and economists debate which of the two developed first, but that's immaterial for our purposes.

In a credit-based system, in the example above, Alice and Bob would be able to trade with each other. Bob would give Alice the tool and Bob gets a favor that's owed to him. In other words, Alice has a debt that she needs to settle with Bob some time in the future. Alice's material needs are now satisfied, but she has a debt that she'd like to cancel, so that's her new "want". If Alice encounters Carol in the future, Alice can trade her food for Carol's medicine, then go back to Bob with the medicine and cancel the debt.

On the other hand, in a cash-based system, Alice would buy the tool from Bob. Later, she might sell her food to Carol, and Carol can sell her medicine to Bob, completing the cycle. These trades can happen in any order, provided that the buyer in each transaction has cash on hand. In the end, of course, it's as if no money ever changed hands.

Neither system is clearly superior. A cash-based system needs to be "bootstrapped" with some initial allocation of cash, without which no trades can occur. A credit-based system doesn't need bootstrapping, but the drawback is that anyone who's owed a debt is taking on some risk. There's a chance that the other person never comes back to settle the debt.

Cash also allows us to be precise about how much something is worth. If you're bartering, it's hard to say if a tool is worth more than medicine or medicine is worth more than food. Cash lets us use numbers to talk about value. That's why we use a blended system today — even when we're using credit, we measure debt in the amount of cash it would take to settle it.

These ideas come up in many contexts, especially online systems where users trade virtual goods of some kind. For example, peer-to-peer file-sharing networks must deal with the problem of "freeloaders," that is, users who download files without sharing in turn. While swapping files might

work, there is also the issue of coordination: finding the perfect person who has exactly the file you want and wants exactly the file you have. In projects like MojoNation and academic proposals like Karma, users get some initial allocation of virtual cash that they must spend to receive a file and earn when they send a copy of a file to another user. In both cases, one or more central servers help keep track of users' balances and may offer exchange services between their internal currency and traditional currency. While MojoNation did not survive long enough to implement such an exchange, it became the intellectual ancestor of some protocols used today: BitTorrent and Tahoe-LAFS.

The trouble with credit cards online

Credit and cash are fundamental ideas, to the point that we can sort the multitude of electronic payment methods into two piles. Bitcoin is obviously in the "cash" pile, but let's look at the other one first.

Credit card transactions are the dominant payment method that is used on the web today. If you've ever bought something from an online seller such as Amazon, you know how the arrangement goes. You type in your credit card details, you send it to Amazon, and then Amazon turns around with these credit card details and they talk to the "system"—a financial system involving processors, banks, credit card companies, and other intermediaries.

On the other hand, if you use something like PayPal, what you see is an intermediary architecture. There's a company that sits between you and the seller, so you send your credit card details to this intermediary, which approves the transaction and notifies the seller. The intermediary will settle its balance with the seller at the end of each day.

What you gain from this architecture is that you don't have to give the seller your credit card details, which can be a security risk. You might not even have to give the seller your identity, which would improve your privacy as well. The downside is that you lose the simplicity of interacting directly with the seller. Both you and the seller might have to have an account with the same intermediary.

Today most of us are comfortable with giving out our credit card information when shopping online, or at least we've grudgingly accepted it. We're also used to companies collecting data about our online shopping and browsing activity. But in the 1990s, the web was new, standards for protocol-level encryption were just emerging, and these concerns made consumers deeply uncertain and hesitant. In particular, it was considered crazy to hand over your credit card details to online vendors of unknown repute over an insecure channel. In such an environment, there was a lot of interest in the intermediary architecture.

A company called FirstVirtual was an early payment intermediary, founded in 1994. Incidentally, they were one of the first companies to set up a purely virtual office with employees spread across the country and communicating over the Internet — hence the name.

FirstVirtual's proposed system was a little like PayPal's current system but preceded it by many years. As a user you'd enroll with them and provide your credit card details. When you want to buy something from a seller, the seller contacts FirstVirtual with the details of the requested payment, FirstVirtual confirms these details with you, and if you approve your credit card gets billed. But two details are interesting. First, all of this communication happened over email; web browsers back in the day were just beginning to universally support encryption protocols like HTTPS, and the multi-party nature of payment protocol added other complexities. (Other intermediaries took the approach of encoding information into URLs or using a custom encryption protocol on top of HTTP.) Second, the customer would have ninety days to dispute the charge, and the merchant would receive the money only after three months! Today the merchant does get paid immediately, but, there still is the risk that the customer will file a chargeback or dispute the credit card statement. If that happens, the merchant will have to return the payment to the credit card company.

In the mid '90s there was a competing approach to the intermediary architecture which we'll call the SET architecture. SET also avoids the need for customers to send credit card information to merchants, but it additionally avoids the user having to enroll with the intermediary. In SET, when you are ready to make a purchase, your browser passes your view of the transaction details to a shopping application on your computer which, together with your credit card details, encrypts it in such a way that only the intermediary can decrypt it, and no one else can (including the seller). Having encrypted your data it this way, you can send it to the seller knowing that it's secure. The seller blindly forwards the encrypted data to the intermediary — along with their own view of the transaction details. The intermediary decrypts your data and approves the transaction only if your view matches the seller's view.

SET was a standard developed by VISA and MasterCard, together with many technology heavyweights of the day: Netscape, IBM, Microsoft, Verisign, and RSA. It was an umbrella specification that unified several existing proposals.

One company that implemented SET was called CyberCash. It was an interesting company in many ways. In addition to credit card payment processing, they had a digital cash product called CyberCoin. This was a micropayment system — intended for small payments such as paying a few cents to read an online newspaper article. That meant that you'd probably never have more than $10 in your CyberCoin account at any time. Yet, amusingly, they were able to get U.S. government (FDIC) insurance for each account for up to $100,000.

There's more. Back when CyberCash operated, there was a misguided — and now abandoned — U.S. government restriction on the export of cryptography, which was considered a weapon. That meant software that incorporated meaningful encryption couldn't be offered for download to users in other countries. However, CyberCash was able to get a special exemption for their software from the Department of State. The government's argument was that extracting the encryption technology out of CyberCash's software would be harder than writing the crypto from scratch.